PB-239 897

ROUTING OF SOLID WASTE COLLECTION VEHICLES

APPENDIX B: OPTIMAL ROUTING OF SOLID WASTE COLLECTION VEHICLES

ILLINOIS UNIVERSITY

PREPARED FOR
NATIONAL ENVIRONMENTAL RESEARCH CENTER

DECEMBER 1974

DISTRIBUTED BY:

NTIS
National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

TECHNICAL REPORT DATA
(Please read Instructions on the reverse before completing)

1. REPORT NO.	2.	3. PB 239 897
EPA-670/2-74-035b		

4. TITLE AND SUBTITLE	5. REPORT DATE
ROUTING OF SOLID WASTE COLLECTION VEHICLES; First Annual Report. Appendix B: Optimal Routing of Solid Waste Collection Vehicles.	December 1974-Issuing Date
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT NO.
Jon C. Liebman and Magne Wathne	
9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT NO.
University of Illinois Urbana, Illinois 61801	1DB314;ROAP 02AAE;TASK 04
	11. CONTRACT/GRANT NO.
	R801289
12. SPONSORING AGENCY NAME AND ADDRESS	13. TYPE OF REPORT AND PERIOD COVERED
National Environmental Research Center Office of Research and Development U.S. Environmental Protection Agency Cincinnati, Ohio 45268	First Annual Report
	14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

See EPA-670/2-74-035a, EPA-670/2-74-036a, and EPA-670/2-74-036b

16. ABSTRACT

The first report provides the background for routing solid waste collection vehicles. Problems encountered with existing methods and possible solutions are given. Appendixes A and B are solutions to two classic routing problems. Appendix A takes the integer programming formulation of Dantzig, Fulkerson, and Johnson which is relaxed to a linear program and a column generation scheme found for its dual. A matching constraint is then added to eliminate some of the fractional solutions which contain an odd cycle of noninteger edges. Finally, a branch-and-bound scheme is employed to search for an optimal all-integer solution. Appendix B is a study applying network theory to the problem of routing a solid waste collection vehicle through a street network. The total length of the tour is sought rather than the detailed and chronologic schedule of the vehicle. This tour is found by minimizing the total length of street segments which have to be retraced. The problem can be described mathematically as a linear program whose optimal solution is guaranteed to be integer. Due to the size of the problem, linear program algorithms cannot be used.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS	b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group
*Collection, *Routing, *Computer programming, Management, Computers, Refuse, *Vehicles, Mathematical models, Linear programming	Solid waste, Solid waste management, *Solid waste collection, Mathematical solutions, *Computer routing	13B

18. DISTRIBUTION STATEMENT	19. SECURITY CLASS (This Report)	21. NO. OF PAGES
Release to public	UNCLASSIFIED	108
	20. SECURITY CLASS (This page)	22. PRICE
	UNCLASSIFIED	5.25

EPA Form 2220-1 (9-73)

Reproduced by
NATIONAL TECHNICAL INFORMATION SERVICE
US Department of Commerce
Springfield, VA. 22151

PRICES SUBJECT TO CHANGE

ABSTRACT

This study is applying network theory to the problem of routing a solid waste collection vehicle through a street network. It is centered on the development of an algorithm to find the minimum tour of a vehicle which has to complete a round trip through a network consisting of both one-way and two-way streets. The work area for the truck is given and its size is presumably determined by the amount of waste generated such that the capacity of the truck is not exceeded. It is assumed that the service rendered by the vehicle can be described as being continuous along each street segment. It is the total length of the tour which is sought rather than the detailed and chronologic schedule of the vehicle. This tour is found by minimizing the total length of street segments which have to be retraced.

The problem can be described mathematically as a linear program whose optimal solution is guarante to be integer. However, the size of the problem does not allow the use of standard

algorithms for solving linear programs. A column generation scheme is developed which operates on the dual problem of the linear program. Each column which is generated and can enter the dual basis, corresponds to the violation of one of the two basic theorems which underlie the algorithm.

EPA-670/2-74-035b
December 1974

071059

ROUTING OF SOLID WASTE COLLECTION VEHICLES

First Annual Report

Appendix B: Optimal Routing of

Solid Waste Collection Vehicles

By

Jon C. Liebman, Principal Investigator,
and Magne Wathne
University of Illinois
Urbana, Illinois 61801

Program Element No. 1DB314

Project Officer

Donald A. Oberacker
Solid and Hazardous Waste Research Laboratory
National Environmental Research Center
Cincinnati, Ohio 45268

NATIONAL ENVIRONMENTAL RESEARCH CENTER
OFFICE OF RESEARCH AND DEVELOPMENT
U.S. ENVIRONMENTAL PROTECTION AGENCY
CINCINNATI, OHIO 45268

w-b

REVIEW NOTICE

The National Environmental Research Center--Cincinnati has reviewed this report and approved its publication. Approval does not signify that the contents necessarily reflect the views and policies of the U.S. Environmental Protection Agency, nor does mention of trade names or commercial products constitute endorsement or recommendation for use.

FOREWORD

Man and his environment must be protected from the adverse effects of pesticides, radiation, noise and other forms of pollution, and the unwise management of solid waste. Efforts to protect the environment require a focus that recognizes the interplay between the components of our physical environment--air, water, and land. The National Environmental Research Centers provide this multidisciplinary focus through programs engaged in

- studies on the effects of environmental contaminants on man and the biosphere, and
- a search for ways to prevent contamination and to recycle valuable resources.

As part of these activities, this report provides the background for routing solid waste collection vehicles, the problems encountered with existing methods, and possible solutions.

A. W. Breidenbach, Ph.D.
Director
National Environmental
Research Center, Cincinnati

ABSTRACT

This study is applying network theory to the problem of routing a solid waste collection vehicle through a street network. It is centered on the development of an algorithm to find the minimum tour of a vehicle which has to complete a round trip through a network consisting of both one-way and two-way streets. The work area for the truck is given and its size is presumably determined by the amount of waste generated such that the capacity of the truck is not exceeded. It is assumed that the service rendered by the vehicle can be described as being continuous along each street segment. It is the total length of the tour which is sought rather than the detailed and chronologic schedule of the vehicle. This tour is found by minimizing the total length of street segments which have to be retraced.

The problem can be described mathematically as a linear program whose optimal solution is guaranteed to be integer. However, the size of the problem does not allow the use of standard

algorithms for solving linear programs. A column generation scheme is developed which operates on the dual problem of the linear program. Each column which is generated and can enter the dual basis, corresponds to the violation of one of the two basic theorems which underlie the algorithm.

PREFACE

The study of techniques for routing solid waste collection vehicles, of which this paper constitutes a portion of the first annual report, is supported by grant EP 801289 of the Solid Waste Research Division of the U. S. Environmental Protection Agency.

This appendix constitutes the doctoral dissertation of Dr. Mågne Wathne, submitted to The Johns Hopkins University. Dr. Wathne is currently Assistant Professor in the Department of Regional Science at the University of Pennsylvania.

Jon C. Liebman
Principal Investigator

TABLE OF CONTENTS

Chapter			Page
1.	INTRODUCTION		1
2.	REVIEW OF LITERATURE		6
	2.1	Heuristic Solutions	8
	2.2	Euler Tours, Chinese Postman's Problem	9
	2.3	Matching	18
3.	THEORETICAL BACKGROUND		23
	3.1	Statement of the Problem	23
	3.2	Feasibility Conditions	26
		3.2.1 Unicursality Theorem	26
		3.2.2 Matching Theorem	30
	3.3	Linear Programming Formulation	31
	3.4	Column Generation	35
		3.4.1 Modified Cut	38
		3.4.2 Blossom Constraints	48
4.	ALGORITHM		54
	4.1	Outline	54
		4.1.1 Format	54
		4.1.2 Major Steps	55
		4.1.3 Geometric Interpretation of Supernodes	57
	4.2	Implementation for Computer	58
		4.2.1 Linear Programming Tableau	58
		4.2.2 Derived Network	61
		4.2.3 Labels and Incidences	63
		4.2.4 Fixed Point Arithmetic	65
		4.2.5 Storage Requirements	71
	4.3	Finiteness	72
	4.4	Relaxation of Some Assumptions	74

Page

5. RESULTS AND DISCUSSION 79

5.1 Computational Results 79
5.1.1 Efficiency of Algorithm 81
5.1.2 Practical Potential . . . 84
5.2 Special Cases 85
5.3 Aspects of Directionality 88

6. SUMMARY AND CONCLUSIONS 90

REFERENCES 97

LIST OF TABLES

Table Page

5.1 Computational Results 80

LIST OF ILLUSTRATIONS

Figure Page

3.1 Network Derived for the Out-of-Kilter Algorithm 49
3.2 Example of Violation of Blossom Constraint 51

CHAPTER ONE

INTRODUCTION

This study is aimed at the application of systems analysis to a problem of routing a solid waste collection vehicle through a street network. The essence of the work is the development of an algorithm to find the minimum tour of a vehicle if the network contains both one-way and two-way streets. Special cases of the problem have previously been considered and solved by network theoreticians who coined the name "The Chinese postman's problem."

Urban management is the combined process of business, politics, and engineering to run a city. Historically, the functions of the city have been considered to be a collection of independent systems. One of the systems is the handling and processing of solid waste. The solid waste system is often broken down into collection, treatment, and final disposal. The assumption here is that the collection process is carried out

by a crew and one of the standard types of collection vehicle. The collection process cannot in the long run be divorced from the design of the treatment and disposal facilities, because the detailed routing of collection vehicles ultimately depends on the location and capacity of those facilities. Other facets of the total problem are: the districting of the city into collection areas, the determination of collection frequency, the choice of type and size of collection equipment.

The present work is solely concerned with the engineering aspects and is conventional in one respect. It isolates one essential part of the problem and ignores all other factors that may affect it. Specifically, the routing of the solid waste collection vehicle is considered as a part of the problem where the treatment and disposal facilities are given. Also known are the collection districts, truck size, and frequency of collection.

The analysis centers on a single truck and its detailed routing through a well-defined network. It is a micro-scale analysis of the process

based on the fundamental assumption that the waste generators are not considered, so the collection service is continuous along each street segment. (This assumption of service provided along the street is also true for street cleaning and snow removal.)

Since all the solid waste of the designated area has to be collected, the total amount collected does not enter into the computations. It is a fixed amount which is limited by the capacity of the truck.

It is also assumed that the truck is making a round trip through its area. This implies that the truck will begin and end its tour at the same point, possibly because that point is closest to the depot. As far as the solution method is concerned, it is arbitrary which point is chosen as the beginning and end point. If the truck were to end its operation at a known destination other than the starting point, then an imaginary street segment would be added between the beginning and end points.

The efficiency of the routing scheme is measured by the total length of the tour. The

time element cannot be included in terms of the chronologic hour of the day. (However, it may be that the measure of length or cost of travel is a weighted combination of distance and time.) Therefore, there is no provision for constraints such as those frequently imposed by the truck crews. They may have implicit understandings or explicit stipulations about time and place for coffee breaks and a lunch hour.

The optimal solution is the minimum length tour such that every street segment is traveled at least once. In order to find a tour that includes every street and also returns to the starting point, it will in general be necessary to travel some streets more than once. The minimum sum of these retracings is in essence what determines the optimal solution.

An optimal solution contains no information about the order of driving the streets. Only the frequency of travel over each one is determined. Existing methods (9) can be used to find the desired route.

The method of analysis employs the same concepts and much of the theory developed for the

earlier versions of the problem which were special cases. Fundamental to the approach is the use of network theory where street intersections are termed nodes, and street segments are called links. The novel feature is the inclusion of both one-way and two-way streets which in network terms become directed and undirected links. Mathematically an undirected link is quite different from a directed link. The directed link behaves precisely like a directed arc in a network flow problem. It will allow flow from one specific node to another specific node. The undirected link has the property that the flow or travel may be in either direction and under some circumstances in both directions. The presence of undirected links therefore makes the problem more combinatorial in nature. The algorithm to handle a mixed network must of necessity contain both aspects, flow characteristics due to directed links and combinatorial characteristics due to undirected links.

CHAPTER TWO

REVIEW OF LITERATURE

Vehicle routing has received increasing attention in the last years as a problem area where systems analysis can be applied. The systematic study of routing problems has been introduced simultaneously with modern analysis techniques of operations research in areas of urban and environmental management. Previous contributions fall under two general classifications. One is the stochastic formulation which often takes the form of a simulation. The purpose is generally to test certain operating procedures and decision rules pertaining to truck size, crew size, collection frequency, etc. Optimization, if done at all, is performed by trial and error, and optimality cannot be guaranteed.

The other type of study, which includes this report, is in the area of optimization models. It usually assumes a deterministic problem and methodically searches among all feasible solutions

for the one which is best, given some measure of effectiveness. In the case of vehicle routing the effectiveness is taken to be the total effort by a vehicle in performing its service. This is most simply measured in time spent or distance travelled which in turn is represented on the street network in terms of lengths of street segments. Basic for the type of service rendered by a solid waste collection vehicle is that it is required to travel every one of the streets in its collection area.

Previous studies of the routing of solid wastes collection vehicles have been either heuristic (to be reviewed in 2.1) or graph and network theoretical. The present work is an addition to the latter category. The underlying graph and network theory has evolved without the motivation that it would ultimately be applied to a practical problem. The oldest predecessor of the vehicle routing problem is the Euler tour problem. The next step is the formulation of the Chinese postman's problem which has turned out to be an acceptable abstraction of the garbage truck problem. The Euler tour and the Chinese postman's

problem will be reviewed in 2.2. The necessary tool for solving the Chinese postman's problem was derived independently and has been termed matching. It is reviewed in the last section.

2.1 Heuristic Solutions

Liebling's suggested solution (15) to the vehicle routing problem is motivated by practical considerations. He considers, as an example, street cleaning which is very similar to garbage collection in providing service along streets. The formulation seeks to find the minimum length of the tour to cover all streets, but also includes finding the location of depots (garages) as well as what machine to employ, and how to partition the network into work areas. His proposed algorithm has several independent components and programs and may be cumbersome to operate on a normal size network.

Bodner, Cassell and Andros (3) suggest a different approach to solve the problem. It simultaneously determines what streets to retrace and the complete route through the network, by simulating the movement of a vehicle through the grid using a random decision process at each street

intersection. The algorithm also includes indicators on when a truck is full and when the work day is over. The resulting solution is a feasible one to the original problem but does not guarantee the optimum solution as pointed out by Marks and Liebman (16). The problem statement differs from the one used in this report in that it does not have an *a priori* specification of what the work area is for one truck.

2.2 Euler Tours, Chinese Postman's Problem

The Chinese postman's problem traces its ancestry back to Euler who in 1736 was given the problem of specifying a walking tour which was to include a given number of bridges exactly once and then return to the origin. The problem has become a classical one in the annals of graph theory. Euler's contribution lay in discovering necessary conditions for such a tour to exist. He modeled bridges as edges and riverbanks and islands as nodes, thus defining a graph. Existence of a tour is guaranteed if the graph is connected and there is an even number of edges touching each node. The practical implication of this requirement is that the number of exits from a node equals the number

of entries. Euler, since called the Father of Graph Theory, was not interested in relaxing the condition of traversing each bridge exactly once to find a minimum length tour which traversed each bridge at least once. His problem was purely combinatorial in nature.

The first descendant of the problem appeared in 1960 when Kwan (14) investigated how to find the walking tour of a mailman. Kwan allows for the possibility that the network is not even at all nodes. His study has since given impetus to further research and also yielded the name, the Chinese postman's problem. The main results are:

a. Each graph has an even number of odd nodes. Each edge adds two incidences to the graph, so the total number of incidences is even. Summed over the nodes there must be an even number of odd incidences for the total to be even.

b. To create a feasible solution to the problem it is necessary to connect distinct pairs of odd nodes with retraced paths. The added incidences

will make odd nodes even thus guaranteeing an Euler tour.

c. The optimum solution is the feasible solution with a minimum total length. It is optimum if the length of retracings on every cycle does not exceed half the length of the cycle.

Kwan's results are insufficient for finding a practical solution algorithm. He suggests starting with a sub-optimal feasible solution and then improving it by finding cycles that violate condition c. This requires enumeration of all cycles in a network, a task whose difficulty grows exponentially.

An efficient and practical algorithm for the Chinese postman's problem was first presented by Edmonds (4). It is based on Kwan's results that a feasible solution is obtainable by the addition of paths between distinct pairs of odd nodes. What made the solution possible was the development of a solution for the matching problem, which finds pairings of nodes in a network so that the sum of intra-pair distances is minimal.

An important point is the fact that we are

only concerned with the retracing of links. No information is needed on the nature of service along the links (i.e., amount of mail or solid wastes), since these are constants of the system and not subject to optimization. Rather, the length of each street segment is the only pertinent information, so the resulting solution is expressed in terms of total distance of retracings. Retracing is the duplication of a link which translates into driving (or walking) a street segment without performing service (solid waste pick-up or mail delivery). Specifying what links to retrace carries no implication about the actual route through the network. It merely says that an optimal Euler tour now exists if the retraced links are considered as duplicates.

The work cited so far contains no reference to direction of travel on any link or street segment. This is an obvious and correct model of Euler's and Kwan's problem in which the travel is on foot unhampered by one-way restrictions. In applying the same technique to vehicle routing, this assumption will in general not hold. In other words, we have considered a completely undirected problem. The equivalent, completely directed

problem has also been solved (17).

The characterization of the directed problem is by means of the balance or unbalance of nodes. A node is balanced if it has the same number of links directed toward it as away from it. An unbalanced node may be deficient if it is incident with more outgoing than incoming links. Likewise, it could be excessive if the opposite is true. A statement can be made about the total unbalance of the graph: the sum of the excesses equals the sum of the deficiencies. An indication of the proof of this statement follows from the observation that each link is outgoing at one node and incoming at another. So, the total of all outgoing incidences for the whole graph equals the sum of all incoming ones. Subtract from each of the two sums those incidences which balance each other at each node. The remaining two sums, still identical, are the sums of deficiencies and excesses.

A feasible solution for this problem is obtained by adding directed paths from excessive to deficient nodes until all nodes are balanced. To find the minimum feasible solution we can make use of the Hitchcock transportation algorithm. We let

excessive nodes be sources, where the excess at each node equals the supply. The deficient nodes are demand points such that the deficiency in each case equals the demand. The arcs connecting the sources and sinks in the transportation problem are found in the graph as the shortest paths from each source to each sink. The optimum solution to the transportation problem will therefore tell which set of paths to retrace in the original graph.

The undirected Chinese postman's problem has also been solved by other techniques. Glover (11) uses the property that no cycle must be retraced over more than one-half its length (14). Each iteration consists of finding the shortest path between an arbitrary pair of odd nodes. The path is added to the graph by reversing the sign on the lengths of every link in the path. A negative cycle in this algorithm would be a cycle where the retracing exceeds half the length, and negative cycles are proved never to appear. Thus in the final solution the retraced links are exhibited as being those with negative lengths.

The difficulty of this scheme is the shortest path routine. Shortest path algorithms

exist for directed graphs with no negative cycles but these cannot be applied to Glover's algorithm. Each link would have to be represented as two oppositely directed one-way links. So, if the length is negative then the two directed links would constitute a negative cycle. This would invalidate the use of any existing shortest path algorithm. Glover has supplied a shortest path algorithm for this problem but it is of an enumerative nature and strongly detracts from the efficiency of the overall algorithm.

Another scheme has been presented by Bellman and Cooke (1). The essential feature of their algorithm is a dynamic programming formulation. The number of stages of the problem equals the number of links in the graph. The process consists of finding the minimum path from a given node such that a specified set of links are included in the path. At each stage the minimum must be found for every combination of starting point and link-set of a given number. For any reasonable size problem this procedure will require a large amount of intermediate storage and computational effort.

The problem of an Euler tour on a mixed

graph was first discussed in the literature by Ford and Fulkerson (10). They investigated the necessary and sufficient conditions for unicursality.[1] Their result is an important basis for the work in this report. The conditions are three-fold (an exact statement of them will be presented in the subsequent chapter):

a. The graph must be connected, i.e., every pair of nodes must have a path connecting them. (The directionality of the individual link does not enter into this condition.)

b. Every node in the graph must be incident with an even number of links.

c. The number and mixture of directed and undirected links incident on each node and on each sub-set of nodes must be such as to allow as many exits from the subset as entries into it.

This theorem has also been established by Johnson (13).

Ford and Fulkerson (10) point out the element of circulation which is required in a unicursal

---

[1] A graph is unicursal if it contains a closed route which includes every link and retraced link exactly once.

graph. Any Euler tour is a closed circuit or cycle in a directed or undirected graph respectively. (This terminology is in conformity with Berge (2).) In general, the tour can be separated into component circuits or cycles such that any node appears at most once in any component. Ford and Fulkerson consequently conclude that one may use circulatory flows to test for the existence of unicursality in the graph.

The solution to the Chinese postman's problem on a mixed graph exists for one special case. Edmonds and Johnson (9) considered a problem where every node is met by an even number of links without regard to the balance of the node. They argue that this is a fairly standard configuration in a street network. If every node is even then the graph can be decomposed into cycles such that every link is contained in exactly one of the cycles. The directionality of the individual cycles is allowed to be inconsistent, i.e., some links may be pointed in opposite directions along the cycle. If the original graph does not contain an Euler tour then additional copies of the links may be added to find a feasible solution. The graph is

infeasible only if part c of Ford and Fulkerson's theorem is violated. Connectedness (part a) is assumed to hold, and the evenness (part b) is already assumed for this special case. Retracings therefore occur only as a result of trying to balance all nodes and sub-sets of nodes. A balanced graph is one where each directed link is part of a distinct cycle. So, the minimum feasible solution for the even graph can be described in flow terms as the minimum circulation which allows a flow of at least one in each directed link. The problem can be solved optimally using the out-of-kilter algorithm.

2.3 Matching

The solution to the matching problem which was instrumental in solving the Chinese postman's problem was developed by Edmonds (5), (6). His two original articles plus one subsequent paper (7) deal with the minimum simple complete matching problem. The network under consideration is one with an even number of nodes, say 2n, and a set of links between the nodes. If the graph is not complete then the missing links are assumed to be of infinite length. A simple matching is a set of

links such that each node is touched by no more than one of these links. A complete matching is a solution where every node is touched by a matching link, so the solution consists of exactly n such links.

The matching problem can be expressed in a zero-one integer linear programming form. Solved as a linear program (LP) the integer solution is only guaranteed when the graph has a special structure. That is, if the graph is bipartite or, equivalently, if the graph contains no cycle with an odd number of nodes, then the constraint matrix of the corresponding LP is unimodular. Unimodularity is a sufficient condition to have an integer solution of the LP. Edmonds found that the non-integer solutions came from odd cycles of nodes. To eliminate these solutions he added one constraint to the problem for every subset containing an odd number of nodes. The resulting convex set has only integer extreme points, thereby enforcing the integer requirement of the problem. Edmonds then developed a labelling algorithm to solve the problem based on the expanded set of constraints.

An important thing to note is the form of

the constraints that Edmonds introduced. For any set of nodes, s, where

$$|s| = 2q_s + 1 \quad (q_s \text{ integer})$$

there can only be q_s links in the matching solution among those links with both endpoints in s. In a later paper Edmonds and Johnson (8) present an equivalent version of the constraint. It says that at least one link must be in the matching solution among those with only one end point in s. The first form is the one on which Edmonds based his algorithm, but the second form is the one employed in this study.

It is worth noting that the completely directed version of the Chinese postman's problem can be solved by using Edmonds' matching algorithm. First, note that whereas the algorithm is developed for the general case of a nonbipartite network, it also solves the special case of a bipartite network. A bipartite network is one where the nodes can be divided into two subsets such that no link has both end points in one subset. The transportation problem can be cast in the form of bipartite matching. Instead of each source node create one

unit source node for each unit of supply. Likewise, substitute unit sinks for the original sinks which have demand greater than one. It was pointed out in a previous paragraph that the total supply equals the total demand after the Chinese postman's problem is transformed to a transportation problem. So, we have an identical number of unit sources and unit demands for the bipartite matching. The links between unit sources and unit sinks are the same shortest paths that were computed for the transportation problem. The matching algorithm will find the minimum matching of sources with sinks. In the directed problem it is possible that some links will be retraced more than once, unlike the undirected problem.

We find that the matching algorithm will solve both the directed and undirected Chinese postman's problem. The tempting conclusion that the Chinese postman's problem on a mixed graph can be transformed into a matching is not true, however. The reason is that the combinatorial character of the undirected problem and the flow character of the directed problem cannot be separated. Note that in the undirected problem the direction of travel on each link is not determined while it is for the

directed problem. In the mixed case it must be expected that the solution will specify the direction of retracing on some of the undirected links. Other undirected links may be retraced without a specified direction. In fact, what we will see in the final solution is a set of retracings which could be separated in two parts. All the directed plus some undirected links with direction specified will make up a set of directed cycles. The rest of the undirected links without directional assignment make up a set of undirected cycles. If we knew *a priori* which of the undirected links would be part of the directed cycles, we could find the minimum directed cycles separately from the undirected cycles. Then we could separately apply matching algorithms to the subproblems.

CHAPTER THREE

THEORETICAL BACKGROUND

3.1 Statement of the Problem

The problem is to find a minimum length tour for a solid waste collection vehicle such that it provides service on all street segments in its assigned work area. The assumption is made that the work area is a connected street network. It is also assumed that every street segment is populated and requires service, and that both sides of the street can be serviced simultaneously. It will be shown later how the last assumptions can be relaxed to make the model more general.

The modeling of the problem rests on the ability to abstract the physical picture into a mathematical one by means of concepts from graph and network theory. Intersections or junctions of two or more streets are called nodes. Street segments are called links. (The terms arc and edge are commonly used for directed and undirected segments respectively. They have, however, been

avoided here in favor of a neutral word.) A node is even if it is incident with an even number of links and otherwise it is odd.

A unicursal graph is one on which there exists a continuous closed tour such that every link is used or travelled exactly once (an Euler tour). A graph which is not unicursal may have some of its links duplicated so the problem is to minimize the necessary duplication to achieve unicursality. In the routing of trucks, this corresponds to finding the minimum length of multiple-travelled street segments such that the truck can provide service along every link and return to its dispatching station or depot.

Two observations are important. One is that in a unicursal graph all nodes are even. In a graph which is not unicursal, the retracing needed to achieve unicursality will consist of paths connecting pairs of odd nodes (4), (14). The other observation is that any route where a vehicle ultimately returns to a starting point, consists of one or more subtours or cycles. If the route of the vehicle were known, it would be apparent that the vehicle has traced a cycle each time it crosses its own path. There is at least one way in which

the complete tour could be decomposed into cycles. Each link or retracing must therefore be a part of a cycle in a feasible tour. An even undirected graph is guaranteed to be unicursal, i.e., it contains an Euler tour (4). An even mixed graph will not contain an Euler tour if there is no feasible set of cycles that include every link and retracing in its given direction.

Using these ideas, an alternate statement of the problem can be formulated as follows. Find the minimum set of retracings that renders all nodes even and incorporates every directed link in a consistently directed cycle.

The Chinese postman's problem can be formulated as a linear programming problem. Let

G = (N,A) = a mixed graph

N = the set of nodes in G

A = the set of links in G

x_{ij} = number of traverses along directed link from node i to node j

y_{ij} = number of traverses along undirected link from i to j or vice versa ($y_{ij} = y_{ji}$)

c_{ij} = length of directed link (i,j)

c'_{ij} = length of undirected link (i,j)

e_i = surplus variable

$d_i = \begin{cases} 1 & \text{if node } i \text{ is odd} \\ 0 & \text{if node } i \text{ is even} \end{cases}$

Then

$$\min \sum_i \sum_j c_{ij} x_{ij} + \sum_i \sum_j c'_{ij} y_{ij} \qquad 3\text{-}1$$

s.t.

$$\sum_j (x_{ij} + x_{ji} + y_{ij}) - 2e_i = d_i,$$

for all nodes 3-2

$$e_i \geq 0; \quad x_{ij}, y_{ij} \geq 1 \quad \text{integer} \qquad 3\text{-}3$$

This formulation has two omissions. One is that our problem is an integer program while the convex set of the LP in general has non-integer extreme points. The other is the requirement that every directed link be contained in a consistently directed cycle. The next section will provide a basis for a way to take these requirements into account explicitly.

3.2 Feasibility Conditions

3.2.1 Unicursality Theorem

Let

s = subset of nodes, $s \subset N$

$\bar{s}$ = the complement of s, $s \cup \bar{s} = N$.

Then, an existence theorem due to Ford and Fulkerson (10) states:

G is unicursal if and only if

a) G is connected

b) $\sum_j (x_{ij} + x_{ji} + y_{ij})$ is even for every node i

c) $\sum_{j \in \bar{s}} \sum_{i \in s} (x_{ij} - x_{ji}) \leq \sum_{j \in \bar{s}} \sum_{i \in s} y_{ij}$, for every s

Part b requires the total sum of entries and exits from a node to be even. Part c says that if a set of nodes, s, has an imbalance in directed links connecting s with the rest of the graph, then this imbalance must be compensated by the undirected links connecting s with $\bar{s}$.

The theorem can be reduced for its two special cases. If the graph is undirected then

$$x_{ij} = 0 \quad \text{for all links, and}$$

part c becomes

$$\sum_{i \in s} \sum_{j \in \bar{s}} y_{ij} \geq 0, \text{ for every s} \qquad 3\text{-}4$$

which is superfluous since $y_{ij} \geq 0$ for all links. Part b becomes

$$\sum_j y_{ij} = 2e_i, \quad \text{for every } i \qquad 3\text{-}5$$

and is therefore necessary and sufficient for a connected graph as discovered by Euler.

If the graph is directed then

$$y_{ij} \equiv 0 \quad \text{for all links,}$$

so, b and c of the theorem reduce to

$$\sum_j (x_{ij} + x_{ji}) = 2e_i, \quad \text{for all } i \qquad 3\text{-}6$$

$$\sum_{j \in \bar{s}} \sum_{i \in s} (x_{ij} - x_{ji}) \leq 0, \text{ for all } s. \qquad 3\text{-}7$$

But, 3-7 must also hold if s and $\bar{s}$ are interchanged, so

$$\sum_{j \in s} \sum_{i \in \bar{s}} (x_{ij} - x_{ji}) = 0, \text{ for all } s \qquad 3\text{-}8$$

For a directed graph 3-8 need not be expressed for subsets of more than one node, i.e., if all nodes

in a directed graph are balanced then all subsets are also balanced. So, 3-8 further reduces to

$$\sum_j (x_{ij} - x_{ji}) = 0, \quad \text{for all } i \qquad 3\text{-}9$$

which satisfies and is stronger than 3-6.

In summary, for the directed graph the theorem reduces to the requirement that every node be balanced. Part c in its general form must hold both with regard to $(s,\bar{s})$ and $(\bar{s},s)$. The constraint for the latter case is

$$\sum_{j\in s} \sum_{i\in \bar{s}} (x_{ij} - x_{ji}) \leq \sum_{j\in s} \sum_{i\in \bar{s}} y_{ij}$$

or

$$\sum_{j\in s} \sum_{i\in \bar{s}} (x_{ji} - x_{ij} + y_{ij}) \geq 0 \qquad 3\text{-}10$$

Interchange indexes i and j;

$$\sum_{i\in s} \sum_{j\in \bar{s}} (x_{ij} - x_{ji} + y_{ji}) \geq 0 \qquad 3\text{-}11$$

Note that $y_{ij} = y_{ji}$, so

$$\sum_{i\in s} \sum_{j\in \bar{s}} (x_{ij} - x_{ji} + y_{ij}) \geq 0 \qquad 3\text{-}12$$

3.2.2 Matching Theorem

Edmonds (5) and Edmonds and Johnson (8) have presented a matching theorem. Let

$u_{ij} = x_{ij} - 1$ = number of retracings of directed link (i,j)

$v_{ij} = y_{ij} - 1$ = number of retracings of undirected link (i,j)

q = set containing odd number of odd nodes

$\bar{q}$ = complement of q, $q \cup \bar{q} = N$

Any integer feasible solution to the matching problem lies on a convex set given by 3-1, 3-2, 3-3, and a set of constraints

$$\sum_{i \in q} \sum_{j \in \bar{q}} (u_{ij} + u_{ji} + v_{ij}) \geq 1, \text{ for all } q \quad 3\text{-}13$$

In other words, for any subgraph, q, which has an odd number of odd nodes, at least one retracing must connect q with the rest of the graph. This set of constraints is often referred to as the blossom constraints. The blossom constraints replace 3-2 of the original LP formulation and part b of the unicursality theorem. The matching theorem states that the convex set of all blossom constraints

has integer extreme points, i.e., x_{ij}, y_{ij}, e_i will be all integer.

3.3 LP Formulation

The constraints based on the two theorems can be incorporated in the LP formulation.

$$\min \sum_i \sum_j c_{ij} x_{ij} + \sum_i \sum_j c'_{ij} y_{ij} \qquad 3\text{-}14$$

s.t.

$$\sum_{i \in s} \sum_{j \in \bar{s}} (x_{ij} - x_{ji}) \leq \sum_{i \in s} \sum_{j \in \bar{s}} y_{ij}$$

$$\text{for all } s \qquad 3\text{-}15$$

$$\sum_{i \in q} \sum_{j \in \bar{q}} (u_{ij} + u_{ji} + v_{ij}) \geq 1, \text{ for}$$

$$\text{all } q \qquad 3\text{-}16$$

$$x_{ij} = u_{ij} + 1 \geq 1 \qquad 3\text{-}17$$

$$y_{ij} = v_{ij} + 1 \geq 1 \qquad 3\text{-}18$$

(3-17 and 3-18: for all (i,j))

$$u_{ij}, v_{ij} \geq 0 \qquad 3\text{-}19$$

Assume there are m_1 directed and m_2 undirected links.

In vector notation, define new variables X and U such that

$$X = \begin{pmatrix} \vdots \\ x_{ij} \\ \vdots \\ \hline \vdots \\ y_{ij} \\ \vdots \end{pmatrix} \qquad \text{and} \qquad U = \begin{pmatrix} \vdots \\ u_{ij} \\ \vdots \\ \hline \vdots \\ v_{ij} \\ \vdots \end{pmatrix}$$

then $U = X - \bar{I}$ or $X = U + \bar{I}$,

where $\bar{I}$ is an $(m_1 + m_2) \times 1$ column vector where each element is 1. Let

$$\bar{c} = \begin{pmatrix} \vdots \\ c_{ij} \\ \vdots \\ \hline \vdots \\ c'_{ij} \end{pmatrix}$$

Define vectors $\bar{a}_s$ and $\bar{b}_q$ each with $(m_1 + m_2)$ elements such that

$$a_{sk} = \begin{cases} +1 \text{ if link } k \text{ is directed from } s \text{ to } \bar{s} \\ -1 \text{ if link } k \text{ is directed from } \bar{s} \text{ to } s \\ +1 \text{ if link } k \text{ is undirected between } s \text{ and } \bar{s} \\ 0 \text{ otherwise} \end{cases}$$

and

$$b_{qk} = \begin{cases} 1 \text{ if link } k \text{ is in } (q, \bar{q}) \\ 0 \text{ otherwise} \end{cases}$$

The formulation becomes

$$\min \quad \bar{c}^1 \, (U + \bar{I})$$

s.t.

$$\bar{a}_s' \, (U + \bar{I}) \geq 0 \quad , \text{ for all } s \qquad 3\text{-}20$$

$$\bar{b}_q U \geq 1 \quad , \text{ for all } q \qquad 3\text{-}21$$

$$U \geq \bar{0} \qquad 3\text{-}22$$

or

$$\min \quad \bar{c}' U \qquad 3\text{-}23$$

s.t.

$$\bar{a}_s' \, U \geq - \bar{a}_s' \, \bar{I} \quad , \text{ for all } s \qquad 3\text{-}24$$

$$\bar{b}_q' \, U \geq 1 \quad , \text{ for all } q \qquad 3\text{-}25$$

$$U \geq \bar{0} \qquad 3\text{-}26$$

Associate dual variables α and β with the primal constraints, and let S be the number of sets s and Q be the number of sets q; then the dual can be written:

$$\max \quad \sum_s (-\bar{a}_s' \cdot \bar{I}) \alpha_s + \sum_q \beta_q \qquad 3\text{-}27$$

s.t.

$$[\bar{a}_1 \cdots \bar{a}_S \bar{b}_1 \cdots \bar{b}_Q] \left| \begin{array}{c} \alpha_1 \\ \vdots \\ \alpha_S \\ \hline \beta_1 \\ \vdots \\ Q \end{array} \right| \leq \bar{c} \qquad 3\text{-}28$$

$$\alpha_s, \beta_q \geq 0 \qquad 3\text{-}29$$

The primal has $m_1 + m_2$ variables and S + Q constraints, i.e., one variable for each link and one constraint for each possible subset of nodes plus one constraint for every odd combination of odd nodes. The dual has S + Q variables and $m_1 + m_2$ constraints. $m_1 + m_2$ is a relatively small number (less than 100 in our applications) but S + Q is a very large number on the order of 2^n where n is the number of nodes in the graph.

There is no practical way that either the dual or the primal can be solved for a realistic problem using the complete constraint matrix. The storage requirement alone would exceed the capacity of most available computers given a network with more than 50 nodes.

In an LP the basis requires storage of a square matrix with dimensions equal to the number of constraints. If the constraint matrix can be implicitly maintained it is preferable to work with the dual since it will allow a much smaller basis than the primal. In the following section will be developed the necessary foundation for working with a dual using a simplex procedure but without requiring a complete tableau of $(m_1 + m_2)$ x

(S + Q).

3.4 Column Generation

A column generation algorithm will be developed using the dual formulation. The algorithm allows a minimum amount of tabulated storage which will be primarily the current basis inverse. The rest of the constraint matrix, i.e., the non-basic columns, will be stored implicitly. This can be accomplished due to the fact that each column corresponds to a primal constraint which, in turn, can be developed given the current solution on the graph itself. The dual column generation, therefore, rests on the possibility of monitoring certain variables associated with the graph and determining what primal constraints may be violated. In general, a primal constraint which holds as an equality is associated with a basic dual variable. A primal constraint which either is violated or holds as a strict inequality is associated with a non-basic dual variable.

In standard LP terms let the following be variables applied to the dual.

B^{-1} = current basis inverse

$x_k - c_k$ = simplex evaluators, k = 1,2,,,,(S+Q).

σ_k = simplex multipliers, k = 1,2,,,, (m_1+m_2)

= $z_k - c_k$ for basic columns

c_k = cost of variable k

where

$z_s = \bar{\sigma}\bar{a}_s$, for all s

$z_{q+S} = \bar{\sigma}\bar{b}_q$, for all q

Also, from 3-27

$c_s = -\bar{a}_s' \bar{1}$, for all s

So, $c_{q+S} = 1$, for all q

$$z_s - c_s = (\bar{\sigma} + \bar{1})\ \bar{a}_s \text{ , for all s} \qquad 3\text{-}30$$

$$z_{q+S} - c_{q+S} = \bar{\sigma}\bar{b}_q - 1 \text{ , for all q} \qquad 3\text{-}31$$

A variable can enter the basis if there is a column, k, such that $z_k - c_k < 0$. A column generation scheme is one where the existence of an entering variable can be determined without keeping a complete simplex tableau in an explicit form.

If a node-set s (or q) exists such that $z_s - c_s < 0$ (or $z_{q+S} - c_{q+S} < 0$) then the corresponding column in the original constraint matrix $\bar{a}_s$ (or $\bar{b}_q$) can be generated. The column as it would have appeared in the current tableau is $\bar{y}_s = B^{-1} \bar{a}_s$ (or $y_{q+S} = B^{-1} \bar{b}_q$).

The details of how to find the variable to leave the basis and the pivoting to find the new tableau are discussed in detail in many texts on linear programming, e.g., Hadley (12). It remains to be shown how a set s or q can be generated from the current LP tableau and the graph itself.

The dual problem has $m_1 + m_2$ columns associated with slack or surplus variables. It is shown in any standard work on LP that the simplex evaluators associated with the slacks are the elements of the row vector $\bar{\sigma}$. Therefore, should slack k be a candidate to enter a basis it will be indicated by $\sigma_k < 0$. The column associated with slack k can be immediately entered. It can therefore be assumed in the discussion of column generation that $\sigma_k \geq 0$, for all k.

3.4.1 Modified Cut

Consider

$$z_s - c_s = (\bar{\sigma} + \bar{1}')\ \bar{a}_s$$

Note that there is one value σ_j for every link j. Assume that σ_j is a measure of capacity on the link j in a sense similar to that used in flow networks. The capacity in a flow sense denotes the number of units of flow which the link can accomodate. Let the capacity in the present problem denote an allowable number of times the link can be traveled. Fix the capacity on a link j at $\sigma_j + 1$. Then $\bar{\sigma} + \bar{1}'$ is the capacity vector for the graph. Given a particular set of nodes s, and the corresponding incidence column, $\bar{a}_s$, the vector product $(\bar{\sigma} + \bar{1}')\bar{a}_s$ defines a special sum of the capacities of the links which separate s and $\bar{s}$. From the definition of $\bar{a}_s$, $(\bar{\sigma} + \bar{1}')\bar{a}_s$ is equal to the sum of capacities of undirected links between s and $\bar{s}$ plus directed links from s to $\bar{s}$ minus directed links from $\bar{s}$ to s. The result is a modified cut, borrowing the notion of a cut from network flow theory.

If the value of the modified cut is negative then the capacity of directed and undirected links from s to $\bar{s}$ is not sufficient to balance the capacity of directed links from $\bar{s}$ to s. Therefore, the primal constraint associated with s is violated. Equivalently, the dual variable associated with s can enter a basis and improve the dual objective function.

If the minimum value of any modified cut is non-negative then there is no corresponding dual variable which can enter the basis. Note that the determination of the minimum modified cut is an LP minimization. Let

A_1 = the set of directed links in G

A_2 = the set of undirected links in G ($A_1 \cup A_2 = A$)

K_{ij} = capacity of link (i,j), i.e., if link k is (i,j) then $K_{ij} = \sigma_k + 1 \geq 1$ since $\sigma_k \geq 0$

$$\Pi_i = \begin{cases} 1 & \text{if node i is in s} \\ 0 & \text{if node i is in } \bar{s} \end{cases}$$

$$\gamma_{ij} = \begin{cases} 1 & \text{if } i \in s \text{ and } j \in \bar{s} \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1 & \text{if } i \in \bar{s} \text{ and } j \in s \\ 0 & \text{otherwise} \end{cases}$$

Consider the LP

$$\min \quad \sum_{(i,j)\in A_1} K_{ij}[\gamma_{ij}-\delta_{ij}] + \sum_{(i,j)\in A_2} K_{ij}[\gamma_{ij}+\delta_{ij}] \qquad 3\text{-}32$$

s.t.

$$\Pi_i - \Pi_j + \gamma_{ij} - \delta_{ij} = 0, \text{ for all links } (i,j) \qquad 3\text{-}33$$

$$\Pi_i \leq 1 \quad , \text{ for all nodes } i \qquad 3\text{-}34$$

$$\Pi_i, \gamma_{ij}, \delta_{ij} \geq 0 \qquad 3\text{-}35$$

It can be shown that the upper bounds on γ and δ follow from the characteristics of the problem and the definition of Π.

First, note that the column vectors, V_γ and V_δ, associated with a pair of variables γ_{ij} and δ_{ij}, are linearly dependent. In fact, γ_{ij} and δ_{ij} occur only once each in the constraint matrix, namely in the same equation.

So $$V_\gamma = -V_\delta$$

Further, if (i,j) is directed then the contribution in the objective function of γ_{ij} and δ_{ij} is

$$K_{ij} \cdot \gamma_{ij} + (-K_{ij})\ \delta_{ij},$$

or dropping the subscripts:

$$K\ \gamma + (-K)\ \delta$$

Introduce the standard LP terminology for simplex evaluators ($z_j - c_j$) and simplex multipliers (σ_j) and let the simplex evaluators for the two columns be

$$z_\gamma - c_\gamma = \bar{\sigma}\ V_\gamma - K$$

and

$$\begin{aligned} z_\delta - c_\delta &= \bar{\sigma}\ V - (-K) \\ &= \bar{\sigma}\ (-V_\gamma) - (-K) \\ &= -(Z_\gamma - c_\gamma) \end{aligned}$$

So, if $z_\gamma - c_\gamma < 0$ then $z_\delta - c_\delta > 0$ and vice versa, and if $z_\gamma - c_\gamma = 0$ then $z_\delta - c_\delta = 0$ so only one of the variables γ and δ can be basic in any solution.

If (i,j) is undirected then the contribution to the objective function is

$$K\gamma + K\delta$$

so

$$z_\gamma - c_\gamma = \bar{\sigma} \cdot V_\gamma - K$$

and

$$\begin{aligned} z_\delta - c_\delta &= \bar{\sigma} \cdot V_\delta - K \\ &= \bar{\sigma} \, (-V_\gamma) - K \\ &= -(\bar{\sigma} V_\gamma - K) - 2K \\ &= -(z_\gamma - c_\gamma) - 2K \end{aligned}$$

Hence, if γ is basic then

$$z_\gamma - c_\gamma = 0 \quad \text{and} \quad z_\delta - c_\delta = -2K < 0$$

since K is a strictly positive capacity. An equivalent argument holds if δ is basic. But in a minimization problem any non-basic variable with negative $z_j - c_j$ will remain non-basic and not enter basis. Consequently, at most one of γ and δ will be basic.

Now consider four cases of the constraint for a link (i,j).

i) $i \in s$, $j \in \bar{s}$

then

$$\Pi_i = 1,\ \Pi_j = 0$$

so

$$\gamma_{ij} - \delta_{ij} = 1$$

hence

$$\gamma_{ij} = 1,\ \delta_{ij} = 0$$

ii) $i \in \bar{s}$, $j \in s$

then

$$\Pi_i = 0\ ,\ \Pi_j = 1$$

so

$$\gamma_{ij} - \delta_{ij} = -1$$

hence

$$\gamma_{ij} = 0,\ \delta_{ij} = 1$$

iii) $i \in s$, $j \in s$

then

$$\Pi_i = \Pi_j = 1$$

so

$$\gamma_{ij} = \delta_{ij}$$

but at most one can be basic from arguments above,

hence

$$\gamma_{ij} = \delta_{ij} = 0$$

iv) $i \in \bar{s}$, $j \in \bar{s}$

then

$$\Pi_i = \Pi_j = 0$$

hence

$$\gamma_{ij} = \delta_{ij} = 0$$

as in case iii.

This proves that γ and δ need not be upper bounded in the LP formulation.

To find the dual, associate variables r_{ij} with 3-33 and p_i with 3-34. Re-write the primal objective function

$$\max \quad \sum_{(i,j)\in A_1} K_{ij}(\delta_{ij} - \gamma_{ij}) + \sum_{(i,j)\in A_2} K_{ij}(-\gamma_{ij} - \delta_{ij}) \qquad 3\text{-}36$$

then the dual becomes

$$\min \quad \sum_i p_i \qquad 3\text{-}37$$

s.t.

$$p_i + \sum_j r_{ji} - \sum_j r_{ij} \geq 0, \text{ for all } i \qquad 3\text{-}38$$

$$\left.\begin{aligned} r_{ij} &\geq -K_{ij} \\ -r_{ij} &\geq K_{ij} \end{aligned}\right\} \quad (i,j)\in A_1 \qquad \begin{matrix} 3\text{-}39 \\ 3\text{-}40 \end{matrix}$$

$$\left.\begin{aligned} r_{ij} &\geq -K_{ij} \\ -r_{ij} &\geq K_{ij} \end{aligned}\right\} \quad (i,j)\in A_2 \qquad \begin{matrix} 3\text{-}41 \\ 3\text{-}42 \end{matrix}$$

The bounds on r_{ij} simplify to

$$r_{ij} = -K_{ij} \quad , \ (i,j) \in A_1 \qquad 3\text{-}44$$

$$-K_{ij} \leq r_{ij} \leq K_{ij} \quad , \ (i,j) \in A_2 \qquad 3\text{-}45$$

where $K_{ij} > 0$.

Add the surplus variable, q_i, for each constraint:

$$p_i - q_i + \sum_j r_{ji} - \sum_j r_{ij} = 0, \text{ for all } i \qquad 3\text{-}46$$

$$q_i \geq 0 \qquad 3\text{-}47$$

The columns corresponding to p_i and q_i are linearly dependent so at most one of them is non-zero for any i.

Interpret r_{ij} as a flow in a link (i,j), i.e., from node i to node j. Then p_i and q_i can be interpreted as excess outflow and excess inflow, respectively at node i. Sum the constraints over all nodes

$$\sum_i p_i - \sum_i q_i + \sum_i \sum_j r_{ji} - \sum_i \sum_j r_{ij} = 0 \qquad 3\text{-}48$$

But

$$\sum_i \sum_j r_{ji} = \sum_i \sum_j r_{ij}$$

so

$$\sum_i p_i = \sum_i q_i \qquad 3\text{-}49$$

hence, total net inflow of all nodes equals total net outflow of all nodes.

A circulatory flow pattern with conservation of flow at all nodes can be achieved by assuming the existence of one supernode (s*) and one supersink (t*). Let s* provide flows to balance the excess outflows and let t* receive flows to balance the excess inflows. It has already been shown (3-49) that these two total flows are identical. Connect t* with a directed arc to s*, and the circulation is established. In order to provide for flow in undirected link (i,j), replace it by two directed arcs (i,j) and (j,i). (The term arc will be used in connection with the modified flow network being discussed in this section. This is consistent with accepted terminology since this network contains only directed elements.) Every directed link (i,j) in the original graph is

replaced by an arc (j,i) in the flow network.

Expression 3-37 can be interpreted as a cost minimization by assigning a unit cost to all variables p_i and zero cost to all q_i and r_{ij}. 3-44 sets the flow equal to K_{ij} on the oppositely directed arc (j,i) replacing a directed link (i,j). That is equivalent to setting the lower bound equal to the capacity which is K_{ij}. 3-45 imposes a capacity of K_{ij} on both the arcs replacing undirected link (i,j) but a lower bound of zero. Let the capacity be infinite and the lower bound equal to zero on all other arcs, i.e., those corresponding to p and q as well as the single arc (t*,s*).

In summary, the LP to find minimum modified cuts is:

$$\min \sum_i p_i \qquad 3\text{-}50$$

s.t.

$$p_i - q_i + \sum_j r_{ji} - \sum_j r_{ij} = 0, \text{ for all } (i,j) \qquad 3\text{-}51$$

$$r_{ij} = K_{ij} \quad , (i,j) \in A_1 \qquad 3\text{-}52$$

$$-K_{ij} \leq r_{ij} \leq K_{ij}, \ (i,j) \in A_2 \qquad 3\text{-}53$$

$$q_i, p_i \geq 0 \qquad 3\text{-}54$$

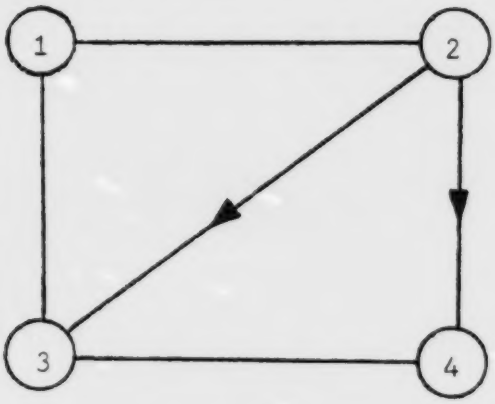
and is interpreted as a minimum cost circulatory flow on a network derived from the original graph. An example of a graph and its derived network are shown in Figure 3.1.[2] The circulatory flow is forced by the lower bound on those arcs which correspond to directed links. The total flow itself can be observed in the arc (t*,s*). A problem of this type is solvable by the OUT-OF-KILTER algorithm (OKA) developed by Ford and Fulkerson (10). The OKA is available as a computer code written in FORTRAN IV and distributed by the Share library.

3.4.2 Blossom Constraints

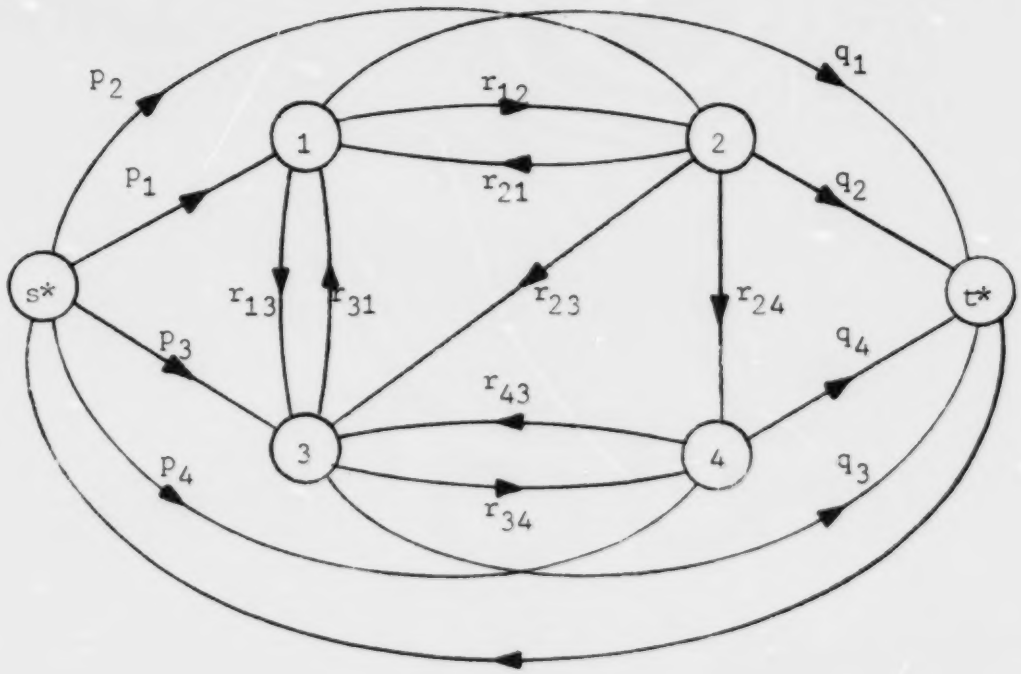
The blossom constraint (3-13 or 3-25) is invoked in four different cases with regard to the primal of the Chinese postman's problem. In the dual, the equivalent column can be entered into the basis. In its simplest form let q be a node set consisting of a single odd node. But the blossom constraint simply says the node must be incident with at least one retracing. This differs somewhat from Kwan (14) who says that on an undirected graph each odd node must be incident

---

[2]The derived network has n+2 nodes and $m_1+2(m_2+n)+1$ arcs.



a. Original Graph



b. Derived Network

Fig. 3.1--Network Derived for the Out-of Kilter Algorithm

with exactly one retraced path. In the mixed graph, however, there may be any positive, odd number of retracings incident at an odd node.

A variation of the first case is obtained by considering an odd node plus a set of even nodes. In effect, applying the blossom constraint to such a set q implies that as long as there is exactly one odd node in q, then q must be connected with at least one retracing. The term supernode will be applied to a set of nodes which contains one odd node. A supernode will be termed active if it is not incident with any retraced path. A single odd node can be considered to be a supernode which contains no even nodes. In the remainder of the discussion the term supernode will be used where this generality is appropriate rather than the more specific term odd node.

A third case calling for blossom constraints arises if an odd number of supernodes are connected in a cycle via non-integer retracings. This is equivalent to what Edmonds (7) called a pseudonode, a term which will be retained in this discussion. Note that total number of nodes in a

pseudonode may be even if it contains an odd number of odd nodes plus an odd number of even nodes. It is therefore also possible that the pseudonode contains an even number of non-integer retraced links. It is the number of non-integer retraced paths between odd nodes which is odd.

A fourth configuration which violates a blossom constraint is one where q is an odd set of supernodes which are connected by retracings but such that there are no retraced paths between q and $\bar{q}$.

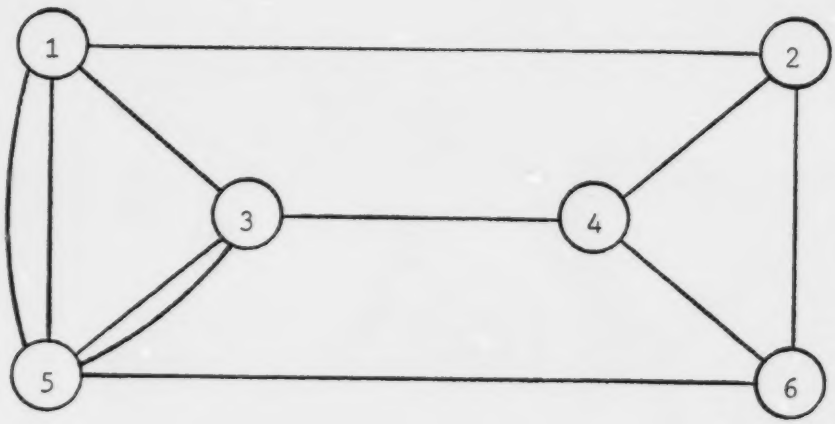


Fig. 3.2--Example of Violation of Blossom Constraint

A simple example is shown in Figure 3.2. The set q consists of nodes 1, 3 and 5 in a simple matching problem. Links (1,5) and (3,5) are retraced. (This situation may occur if the triangle inequality does not hold so that the length of (1,5) plus (3,5) is less than the length of (1,3).)

In general, if q is odd and all supernodes in q are incident with at least one retraced path, then the total number of incidences between paths and supernodes is even (two incidences for each path). But this sum must equal the number of incidences summed over the supernodes in q. If an odd number of non-zero incidences make an even sum, there must be an odd number of even incidences plus an even number of odd incidences. Therefore, an odd number of supernodes in q have an even incidence of retraced links. Let q be a pseudonode and apply a blossom constraint on q.

In Figure 3.2 node 5 is even in retraced links. So, the blossom constraint will require at least one retracing between nodes 1,3,5 and nodes 2,4,6.

In order to generate the blossom constraints (3-25) the incidence vector $\bar{b}_q$ can be

found given the node set q. But $\bar{b}_q$ is the original column which corresponds to a variable that can be entered in the dual tableau. The column, as it would have appeared in the current tableau, is

$$\bar{y} = B^{-1} \cdot \bar{b}_q$$

A column in the dual resulting from a blossom constraint may enter the basis in the same way as a column generated by the OKA.

CHAPTER FOUR

ALGORITHM

4.1 Outline

4.1.1 Format

Recall the linear programming formulation

$$\min \quad \bar{c}'U \tag{4-1}$$

s.t.

$$\bar{a}_s'U \geq -\bar{a}_s\bar{1} \quad , \text{ for all } s \tag{4-2}$$

$$\bar{b}_q'U \geq 1 \quad , \text{ for all } q \tag{4-3}$$

$$U \geq \bar{0} \tag{4-4}$$

and its dual

$$\max \quad \sum_s(-\bar{a}_s'\bar{1})\ \alpha_s + \sum_q \beta_q$$

s.t.

$$[\bar{a}_1\bar{a}_2 \cdots \bar{a}_S\bar{b}_1\bar{b}_2 \cdots \bar{b}_Q] \begin{pmatrix} \alpha_1 \\ \beta_2 \\ \vdots \\ \alpha_S \\ \beta_1 \\ \vdots \\ \beta_Q \end{pmatrix} \leq \bar{c} \tag{4-6}$$

$$\alpha_s,\ \beta_q \geq 0 \tag{4-7}$$

The algorithm basically follows a simplex format, operating on the dual problem. It has a large number of variables, and to each variable corresponds a column in the (dual) convex set. At any iteration the variable which is a candidate to enter the basis will be generated outside the tableau. Only the inverse basis plus columns generated by blossom constraints are included in the tableau.

The algorithm has three functions:

1. To produce a simplex tableau for every iteration including the dual and primal variables plus simplex multipliers and evaluators. These are all standard simplex operations.
2. To keep track of the original network with incidences and labels on supernodes and pseudonodes.
3. To generate columns for entrance into basis.

4.1.2 Major Steps

Step 0

Initialize linear programming tableau.

Set up the OKA network.

Find initial odd nodes.

Initialize arrays.

If graph is even, go to step 4.

Otherwise, go to step 2.

Step 1

Enter all explicit columns whose $z_j - c_j$ is negative.

Go to step 2.

Step 2

Grow supernodes until no supernode is active.

Go to step 3.

Step 3

Enter all explicit columns whose $z_j - c_j$ is negative.

If at least one column was entered, go to step 2.

Otherwise, if network is undirected, go to step 5.

Otherwise, go to step 4.

Step 4

Find the minimum modified cut by the OKA.

If it is non-negative, go to step 5.

Otherwise, generate and enter its corresponding column, and go to step 1.

Step 5

Search for a non-integer retracing.
If none exist, go to step 6.
Otherwise, find the pseudonode, generate and enter the corresponding column, and go to step 1.

Step 6

Search for a supernode incident with an even number of retracings. If none exist, go to step 7.
Otherwise, find the pseudonode, generate and enter the corresponding column, and go to step 1.

Step 7

Output. Terminate with optimal solution.

4.1.3 Geometric Interpretation of Supernodes

Let an original odd node, r_1, be designated the root of a supernode. All subsequent even nodes included in the supernode, are labeled r_1. First path: Find the closest node (j_1) from the root. The link (r_1, j_1) is in the solution as a retracing. Second path: Find the second closest node, j_2. The path (r_1, j_2) is in the solution

and (r_1, j_1) is not unless it is a part of (r_1, j_2). The supernode is built by successively adding the nearest node. As long as the nearest node is even, the supernode is active. At some stage, it may happen that the nearest node is part of another supernode, r_2. r_1 will then no longer be active since it is now connected with another supernode. If r_2 was active, it too would become inactive. However, it may be that r_2 already was inactive and therefore, becomes connected with more than one supernode.

When an odd number of supernodes are adjacent and form a cycle, a blossom may be formed by connecting the supernodes in the cycle with non-integer retraced links. When a blossom exists, and no column can be generated by the OKA or by extending a supernode, then a pseudonode is created.

4.2 Implementation for Computer

4.2.1 Linear Programming Tableau

The initial basis of a linear program contains only the slack variables whose columns are unit vectors, so the initial inverse basis is an identity matrix. Its dimensions are $(m_1 + m_2)$ x

$(m_1 + m_2)$. When a variable is basic, it corresponds to the primal constraint holding at equality. When the same variable becomes non-basic, its simplex evaluator $z_j - c_j$ will initially be non-negative and the primal constraint holds as an inequality. The non-basic variable may subsequently become a candidate to enter the basis again if its evaluator becomes negative. The negative $z_j - c_j$ corresponds to a primal constraint which is violated.

The columns which are generated in the algorithm fall under two categories which must be treated differently; those which are generated by blossom constraints and those which are generated by the OKA. Those which are generated by blossom constraints must be retained in the tableau to ensure that all previous blossom constraints are satisfied. In order to extend a supernode, there must be a continuous path from the root of the supernode to some adjacent node not in the supernode. But the path may not be continuous unless all blossom constraints which form the supernode are valid. The tableau must be expanded by one column for every blossom constraint. As the

variable enters, a unit column is appended to the tableau with a simplex evaluator equal to 0. When the variable becomes non-basic, the value $z_j - c_j$ will indicate directly whether or not the variable should enter the basis again.

The other category of columns are those which are generated by the OKA. They need not be referred to again even if one of them at some later iteration would have been a candidate to enter basis. The minimum modified cut is an indicator of a lacking return capacity from one part of the network to the other. The algorithm will always generate that partition $(s,\bar{s})$ which most strongly violates this type of continuity. Nowhere in the subsequent iterations (except in the final solution) does the algorithm logically require that the previous OKA-cuts must be satisfied. Therefore, the columns generated by the OKA may be omitted from the tableau if those variables become non-basic again.

Subroutine NONBAS checks $z_j - c_j$ for all existing columns in the tableau. If any one is negative, then that column will re-enter the basis. The departing column is found and pivoting is

carried out in the usual way, but no new column is added to the tableau since it already is included. NONBAS will re-enter all columns with $z_j - c_j < 0$. It is called before growing supernodes or before finding a modified cut. This insures that all blossom constraints hold before generating any other column. In the case of the OKA, all capacities must be positive; therefore, all simplex multipliers must be non-negative.

4.2.2 Derived Network

The flow network used in determining the minimum modified cut is derived from the original network as sketched in section 3.4.1. Every directed link in the original network corresponds to an arc in the derived network and every un-directed link corresponds to a pair of oppositely directed arcs. Each time a minimum modified cut is sought, there is a transfer of information from the linear programming tableau to the OKA network by assigning capacities and lower bounds to the arcs. The array LARC (j) provides the reference. For a directed link, $j \leq m_1$, LARC (j) corresponds to the arc. For an undirected link, $m_1 < j \leq m_1 +$

m_2, there are two arcs, LARC (j) and LARC $(m_1 + m_2 + j)$. As previously shown, the capacities in the derived network are determined from the vector $\bar{\sigma}$, which must be non-negative. All those arcs of the network with at least one end point incident at the super source s* or supersink t*, have constant capacities and lower bounds which are set once at the initialization of the program and stay constant thereafter.

GENRTR will call the OKA to compute the minimum modified cut. The OKA will only find a non-zero minimum cut if it is strictly negative, in which case there is a column that can be entered. The set s can be found directly from the dual variables at the termination of the OKA. The dual variables take on one of two values. The set $\bar{s}$ plus the super-source s* and the supersink t* are at one value, say Π, and the set s at another, $\Pi + 1$. Given the set s, the incidence column $\bar{a}_s$ is given by

$$a_{sk} = \begin{cases} +1 & \text{if link k is directed from s to } \bar{s} \\ -1 & \text{if link k is directed from } \bar{s} \text{ to s} \\ +1 & \text{if link k is undirected between s and } \bar{s} \\ 0 & \text{otherwise} \end{cases}$$

Then $\bar{y}$ is given by

$$\bar{y} = \bar{B}^{-1}\bar{a}_s$$

where $\bar{B}^{-1}$ is the inverse basis.

4.2.3 Labels and Incidences

The information about the network necessary to check blossom constraints is stored in several arrays. The arrays change as functions of the primal solution, i.e., retracings. The primal variables are the dual variables of the dual problem, hence the vector $\bar{\sigma}$ represents retracings. The values in $\bar{\sigma}$ change during each simplex iteration, so the updating of the network is a subsidiary function of pivoting, which is carried out in sub-routines PIVOT and SIGLUP.

Supernodes are grown irreversibly and are labeled from the respective original odd nodes. An unlabeled even node is labeled when it is connected with a supernode via a link which is retraced at least once. These labels are stored in LABL (i).

The array SUPER (i) shows how many retraced paths are incident with any supernode i.

If SUPER (i) = -1, then i is an active supernode, i.e., odd. If SUPER (i) = x where $x \geq 0$, then supernode i is incident with x + 1 retraced links.

The link-node incidence of the network is stored in two arrays, NODI (j) and NODJ (j). The former is the source and the latter is the sink if j is a directed link. If j is undirected then the two variables are the end points of the link.

NSUPR (i,j) records supernodes and pseudonodes and their connecting retraced paths. If node j is in supernode or pseudonode i, then NSUPR (i,j) = 1. If nodes i and j are in different supernodes and are the end points of a retraced link, then NSUPR (i,j) = -1 and NSUPR (j,i) = -1. The NSUPR-matrix can therefore be used to find all nodes in a supernode as well as all other supernodes that are connected through retraced paths.

The array LABT (i) contains the top-of-the-nest pseudonode, i.e., the largest and latest supernode which contains node i.

The total incidence on individual nodes is kept in the array DGRE (i). The array has no

function in the algorithm and is purely for display of intermediate results.

4.2.4 Fixed Point Arithmetic

The algorithm as outlined so far allows for non-integer intermediate solutions. Non-integer tableaus may occur since the convex set is not unimodular. The final solution, however, must be integer, so a certain measure of precision must be maintained throughout the computation. In a large example inescapable round-off errors may occur due to floating point arithmetic of the computer. If all the numbers generated were presumed integer, then the round-off errors may be no obstacle at all. However, proper non-integer numbers are subject to the same errors but cannot be checked. In order to achieve maximum control of the accuracy, the variables in the computer code are multiplied by a common denominator and kept in integer form. This scheme facilitates the use of fixed point arithmetic and therefore escapes the round-off error.

The derivation, as shown in the previous chapter, must be modified to take account of the scale factor involved. First, four prerequisites

must be established.

1. All input parameters must be integer.

If the primal problem is written as

$$\min \bar{c}'U \qquad 4\text{-}8$$

s.t.

$$AU \geq E \qquad 4\text{-}9$$

$$U \geq 0 \qquad 4\text{-}10$$

then $\bar{c}$, A, and E must be integer. This is clearly true for A which is an incidence matrix and E which consists of zeros and ones. So, all lengths of street segments, $\bar{c}$, must be provided as integers.

2. The determinant of the basis will clear all fractions, i.e., solution vectors are rational numbers and the determinant is a common denominator. Given a square matrix D and vectors x and E such that

$$Dx = E \qquad 4\text{-}11$$

where D and E are integer. If D is non-singular, then

$$x = \frac{1}{|D|} (\text{adj } D) \cdot E \qquad 4\text{-}12$$

where D is the determinant of D and adj D is the adjoint of D. Note that

$$\text{adj } D = |D| \cdot D^{-1}. \qquad 4\text{-}13$$

$|D|$ and adj D are integer since D is integer. Then

$$|D|\ x = (\text{adj } D) \cdot E \qquad 4\text{-}14$$

where E is integer, hence (adj D)·E is integer, therefore $|D|$ x is integer.

3. In a linear program the determinant of the current basis is the product of all the previous pivot elements.
For a verification of this statement the reader is referred to a book on linear programming (i.e., Hadley's (12) discussion on product form of the inverse).

4. The initial basis has a determinant equal to 1. Initially the basis is an identity matrix. The

determinant of an identity matrix is always one.

Let the following be the unscaled variables as discussed previously in this report.

b = current basis, $(m_1+m_2)x(m_1+m_2)$

b^{-1} = inverse of b, $(m_1+m_2)x(m_1+m_2)$

b_{ij} = element of b^{-1}

σ_j = element of $\bar{\sigma}$ or $z_j - c_j$

z-c = the value of "$z_j - c_j$" for the entering column

y_j = element of $\bar{y}$

where

$$\bar{y} = b^{-1} \cdot \bar{a}$$

and $\bar{a}$ is either a column $\bar{a}_s$ or $\bar{b}_q$. Let B, B^{-1}, B_{ij}, G_j, Z, and Y_j be the scaled up version of the variables. (In the computer code the representation of the last four is BINV (i,j), SIGMA (j), ZJCJ, and YS (j).) Note that the incidence column $\bar{a}$ is integer by definition (0,1,-1) and need not be changed. (The incidence column is coded as AS (j) regardless of whether it

represents $\bar{a}_s$ or $\bar{b}_q$.)

$$B^{-1} = |b| \cdot b^{-1}$$

as pointed out in prerequisite 2, so

$$B_{ij} = |b| \cdot b_{ij} \qquad 4\text{-}15$$

Therefore,

$$y_j = \sum_{j=1}^{m_1+m_2} b_{ij} a_j$$

$$= \frac{1}{|b|} \sum_{j=1}^{m_1+m_2} B_{ij} a_j \qquad 4\text{-}16$$

hence, let

$$Y_j = |b| \cdot y_j \qquad 4\text{-}17$$

The standard procedure for determining what variable leaves basis, yields the pivot element y_ℓ. According to prerequisite 3 the new determinant $|\hat{b}|$, is given by

$$|\hat{b}| = b \cdot y_\ell \qquad 4\text{-}18$$

but

$$y_\ell = \frac{Y_\ell}{|b|} \qquad 4\text{-}19$$

so

$$|\hat{b}| = Y_\ell \qquad 4\text{-}20$$

The pivoting of the tableau is governed by

$$\hat{b}_{ij} = b_{ij} - b_{\ell j} \frac{y_i}{y_\ell} \quad \text{for } i \neq \ell \qquad 4\text{-}21$$

or

$$\frac{\hat{B}_{ij}}{|\hat{b}|} = \frac{B_{ij}}{|b|} - \frac{|B_{\ell j}|}{|b|} \cdot \frac{Y_i \cdot |b|}{|b| \cdot Y_\ell} \qquad 4\text{-}22$$

but

$$|\hat{b}| = Y_\ell \qquad 4\text{-}23$$

so

$$\hat{B}_{ij} = \frac{B_{ij} \cdot Y_\ell - B_{\ell j} \cdot Y_i}{|b|} \text{ for } i \neq \ell \qquad 4\text{-}24$$

Also

$$\hat{b}_{\ell j} = \frac{b_{\ell j}}{y_\ell} \qquad 4.25$$

or

$$\frac{\hat{B}_{\ell j}}{|\hat{b}|} = \frac{B_{\ell j} \cdot |b|}{|b| \; Y_\ell} \qquad 4\text{-}26$$

so

$$\hat{B}_{\ell j} = B_{\ell j} \qquad 4\text{-}27$$

Likewise

$$\hat{\sigma}_j = \sigma_j - \frac{(z-c)\cdot b_{\ell j}}{y_\ell} \qquad 4\text{-}28$$

so

$$\hat{G}_j = \frac{G_j \cdot Y_\ell - Z \cdot B_{\ell j}}{|b|} \qquad 4\text{-}29$$

The dual variables α_i, have not been multiplied by $|b|$ since the product $\alpha_i|b|$ may exceed the allowable value for an integer variable in the computer. The value of α_i does not influence the accuracy of $\bar{\sigma}$ directly, although it may affect the choice of the departing vector.

The variable SUPER (i) for any supernode i, records the cumulative change in retracings on all links incident with i and is similarly multiplied by the scale factor.

4.2.5 Storage Requirements

The main part of the computer code requires the storage of six vectors associated with the nodes and six with the links. In addition, one vector (SIGMA (j)) is dimensioned at $2(m_1+m_2)$ to allow for $\bar{\sigma}$ plus all the z_j-c_j's of the columns

generated from blossom constraints. The main tableau (BINV (i,j)) is (m_1+m_2)x $2(m_1+m_2)$, also to allow for as many as m_1+m_2 extra columns. The array in SUPER (i,j) seems sufficiently dimensioned at approximately 1 1/2 nxn. One vector (LARC (j)) must accomodate the number of links in the derived network i.e., $m_1+2(m_2+n)+1$ elements.

The set of subroutines for the OKA have an additional requirement of six arrays on the nodes and seven on the arcs. The total need for memory of arrays is therefore

$$2(m_1+m_2)^2 + 1\ 1/2n^2 + 28n + 16m_1 + 24m_2.$$

4.3 Finiteness

Arguments about finiteness are basically the same as those that are given for any standard linear programming problem. If the whole convex set were given, the number of columns (S + Q) would be finite so the formulation does not differ in that regard from a standard problem. The more sophisticated methods of preventing cycling have not been included in the present code and therefore represent a possible pitfall. However, cycling has

not occurred in any of the examples considered in this study, and experience has shown that cycling is an extremely rare occurrence in any linear program.

Some specific considerations can be made about the finiteness of columns generated from blossom constraints. A blossom constraint on any supernode or pseudonode can only be introduced if it is incident with less than one retracing. Also, any blossom constraint can only generate a column once. The reason is that all previous columns from blossom constraints are stored in the tableau and are searched for possible entering variable before making another pass through active supernodes. Finally, $\bar{s}$ must be non-empty such that there is at least one link in $(s,\bar{s})$. Since each column entered adds one retraced link to the network, the maximum number of possible blossom constraints equals the number of links (m_1+m_2).

One crucial assumption has been made about the structure of the problem. It is expressed as connectedness in the unicursality theorem by Ford and Fulkerson (10), but in practical terms it means that a feasible solution exists. The justification

for this assumption is that the networks considered are models of real street networks. The systems analyst (indeed the whole public) must take it for granted that he can reach any point on the network from any other point traveling by car. The implication for the algorithm is that a feasible solution can be found, and it represents an upper bound for the optimum solution. One upper bound is the tour traced out by the truck itself if it already is employed in the specified network.

4.4 Relaxation of Some Assumptions

The modeling and analysis have been done under the following assumptions:

1. Both sides of the street can be serviced simultaneously from either side.
2. All street segments require service.

The assumption of simultaneous serivce of both sides does not always hold. In the case of a multi-lane street or a major artery the collection will be carried out on one side at a time. To incorporate this situation into the algorithm, one simply creates two one-way links, i.e., one for each side of the street. If the street is

one-way the two links have the same direction. If it is a two-way street the two links are oppositely directed.

If the second assumption is relaxed, it allows the modeling of street segments which do not require service. Examples are superhighways, uninhabited streets, and streets outside the work area of the truck. If these streets are available for travelling by the truck, they can be included in the problem.

Let

w_{ij} = number of traverses along directed link from i to j

z_{ij} = number of traverses on undirected link (i,j) - $(z_{ij} = z_{ji})$

where link (i,j) in both cases requires no service. The problem statement 3-1, 3-2, 3-3, can be amended as follows.

$$\min \sum_i \sum_j (c_{ij}x_{ij} + c_{ij}w_{ij} + c'_{ij}y_{ij} + c'_{ij}z_{ij}) \qquad 4\text{-}30$$

s.t.

$$\sum_j (x_{ij}+x_{ji}+w_{ij}+w_{ji}+y_{ij}+z_{ij}) - 2e_i = d_i, \text{ for all } i \qquad 4\text{-}31$$

$$x_{ij},\ y_{ij} \geq 1 \qquad 4\text{-}32$$

$$e_i,\ w_{ij},\ z_{ij} \geq 0 \qquad 4\text{-}33$$

Part c of the unicursality theorem can be applied as before;

$$\sum_{i \in s} \sum_{j \in \bar{s}} (x_{ij} - x_{ji} + w_{ij} - w_{ji} + y_{ij} + z_{ij}) \geq 0, \text{ for all } s \qquad 4\text{-}34$$

The matching theorem can also be applied. It is a summation of non-required travel, so

$$\sum_{i \in q} \sum_{j \in \bar{q}} (u_{ij} + u_{ji} + w_{ij} + w_{ji} + y_{ij} + z_{ij}) \geq 1, \text{ for all } q \qquad 4\text{-}35$$

Introduce vector notation by defining new variables X, U, and a constant vector L, such that

$$X = \begin{pmatrix} \vdots \\ x_{ij} \\ \vdots \\ \hline \vdots \\ y_{ij} \\ \vdots \\ \hline \vdots \\ w_{ij} \\ \vdots \\ \hline \vdots \\ z_{ij} \\ \vdots \end{pmatrix} \quad U = \begin{pmatrix} \vdots \\ u_{ij} \\ \vdots \\ \hline \vdots \\ v_{ij} \\ \vdots \\ \hline \vdots \\ w_{ij} \\ \vdots \\ \hline \vdots \\ z_{ij} \\ \vdots \end{pmatrix} \quad L = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \\ \hline 0 \\ 0 \\ \vdots \\ 0 \\ \hline 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad 4\text{-}36$$

Then

$$X = U + L \qquad 4\text{-}37$$

In the new notation the problem becomes

$$\min \ \bar{c}' U \qquad 4\text{-}38$$

s.t.

$$\bar{a}_s' U \geq -\bar{a}_s' L \quad , \text{ for all } s \qquad 4\text{-}39$$

$$\bar{b}_q' U \geq 1 \quad , \text{ for all } q \qquad 4\text{-}40$$

$$U \geq \bar{0} \qquad 4\text{-}41$$

Take the dual,

$$\max \ \sum_s (-\bar{a}_s' L)\, \alpha_s + \sum_q \beta_q \qquad 4\text{-}42$$

s.t.

$$[\bar{a}_1 \bar{a}_2 \cdots \bar{a}_S \bar{b}_1 \bar{b}_2 \cdots \bar{b}_Q] \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_S \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_Q \end{pmatrix} \leq \bar{c} \qquad 4\text{-}43$$

$$\alpha_s, \beta_q \geq 0 \qquad 4\text{-}44$$

The variable entering the dual basis is determined

by

$$z_s - c_s = (\bar{\sigma} + L') \bar{a}_s \qquad 4\text{-}45$$

$$z_q - c_q = \bar{\sigma} \cdot \bar{b}_q - 1 \qquad 4\text{-}46$$

The difference shows up in $z_s - c_s$ since some elements in L are zero. The capacity in the derived network is $\sigma_j + 1$ for all links, j, with service, and σ_j for all links, j, without service. Therefore, the difference between streets that require service and those that do not must be recognized at the point in the algorithm where the input is provided for the OKA. The blossom constraints are monitored as before, i.e., there is no change in the procedure for finding $z_q - c_q$. The algorithm proceeds as before except for minor modifications.

CHAPTER FIVE

RESULTS AND DISCUSSION

5.1 Computation Results

The algorithm was coded in FORTRAN IV and tested on eight different problems on an IBM 7094 computer. Table 5.1 lists the main descriptions of the eight problems and some characteristics of the solutions.

The algorithm requires at least one directed link and one undirected link. If the network is completely directed, it is necessary to find a node which is incident with exactly two links, and remove the directional assignment on one of them. In the case of an undirected network, one may arbitrarily assign a direction to one link without loss of generality.

There is one other requirement on the network. A directed dead-end street will create a non-feasible solution but should not be expected to appear if the system to be modeled is a street network. The algorithm will handle dead-end streets provided they are undirected. It recognizes a dead-end as an odd node which must be made the end point of a retraced path. Since part of the retraced path must fall along the dead-end street, the link and end-node can be ignored in the data.

TABLE 5.1

COMPUTATIONAL RESULTS

a. Description of Problems

Problem Number	Nodes	Directed Links	Undirected Links	Original Odd Nodes	Arcs in Derived Network
1	6	1	8	6	30
2	6	3	6	6	28
3	11	4	14	4	55
4	16	9	21	10	84
5	46	1	74	30	242
6	46	43	32	30	200
7	38	1	60	22	198
8	38	29	32	22	170

b. Characteristics of Solutions

Problem Number	Columns by OKA	Columns by Blossom Constraints	Retracings	Linear Programming Iterations	Execution Time (min.)
1	0	7	3	7	.02
2	2	6	5	12	.02
3	0	8	6	8	.02
4	2	16	6	22	.06
5	0	43	22	46	.24
6	37	40	36	284	3.13
7	0	31	14	33	.11
8	10	34	22	103	.59

Nodes of degree two, i.e., nodes incident with exactly two links, can be ignored in the algorithm. The two adjacent links can be combined into one with a length equal to the sum of the two. If at least one of the two is directed, then the combined link must also be directed accordingly. The exception to this case is if the network is directed, and one of the two links is made undirected as noted above.

5.1.1 Efficiency of Algorithm

Efficiency in this discussion is measured mainly in terms of execution time. First, the algorithm should be compared to existing algorithms for the special cases of a fully directed or a fully undirected network. Though completely developed theoretically they are not readily available in coded form. It is reasonable to expect, however, that the algorithm for a mixed network will be relatively inefficient applied to the special networks. It will only to a very limited extent take advantage of the types of structure that emerge when the network approaches one of the extreme cases. The directed network uses the transportation algorithm, and the

undirected network employs the matching algorithm. Both of these are very rapid routines which are based on labeling algorithms on the network. By comparison, the mixed network algorithm must handle the general case by means of large amounts of information in the form of arrays which must be up-dated continuously. Each linear programming iteration involves a great deal of computation effort.

Secondly, it is of interest to determine what characteristics of the problem affect the relative efficiency. The results in Table 5-1 seem to indicate that the important factors are the number of nodes, the relative mixture of directed versus undirected links, and the number of initial odd nodes. The effort clearly depends on the size of the network defined by the number of nodes and the number of links. The number of links in most networks modeled after streets, is roughly proportional to the number of nodes. (E.g., $m_1+m_2 = 2n$ for a rectangular network of infinite extent.) Therefore, the number of nodes alone may be a sufficient indicator of size. The ratio of directed to undirected links must bear some

relationship to the number of columns generated by the OKA. If $m_1 = 0$, then no columns will be generated by OKA. If the number of directed links is increased, then the possibility increases of finding minimum modified cuts which can be violated. The number of initial odd nodes affects the number of possible supernodes and therefore the effort involved in growing the supernodes. This point is not clearly borne out by the computational results, so this particular relationship may be weak.

Summarizing, the indications are that execution time is an increasing function of $n, \frac{m_1}{m_2}$ and number of initial odd nodes. The results are not sufficient to establish whether relative maxima exist with respect to $\frac{m_1}{m_2}$ and number of odd nodes for sufficiently high values.

The largest network that was solved had 46 nodes and 75 links. The code was dimensioned to accomodate this network if all its links were undirected. The algorithm occupies a large portion of the available core storage. In a 32,000 word machine there are less than 1,000 unused memory locations. Some saving can probably be realized

through a close scrutiny of the computer code. It is likely that the storage requirement can be somewhat reduced by more efficient allocation, but it will not substantially increase the maximum size of network.

5.1.2 Practical Potential

Problem 6 in Table 5-1 is modeled after an actual area in Baltimore, Maryland. (Problem 5 is the same network made undirected by ignoring all but one of the one-way streets.) The area is approximately 0.2 square miles. The population is 4,183 according to the 1970 census (20). Most of the residences are individual homes and row houses. Truitt et al. (19) say that the average generation of residential solid waste in Baltimore is 1.95 lbs/cap/day. Using this figure and assuming the pick-up takes place twice weekly the truck must collect 29,000 lbs per collection. But a standard size truck (such as one used in Baltimore) normally takes only about 10,000 lbs, so on that basis the area would be divided into three truck loads.

The conclusion is that the capacity of the algorithm is sufficient to handle a realistic size

network. The determining factor of size of the network is the number of nodes, so the population density should therefore be measured in population per node or possibly population per street segment. The present exercise has shown that the algorithm can solve for a truck load on a network with 1/3 the popularion density of a particular Baltimore neighborhood.

5.2 Special Cases

Three special cases will be considered: all nodes are even, all links are directed, all links are undirected. If the network is even, then columns can only be generated by the OKA. There are no initial odd nodes, hence no supernodes and therefore no blossom constraints to grow supernodes.

Note that if every node is even, every subset, s, is even. The partition $(s,\bar{s})$ contains an even number of links, i.e., if it contains e links directed from s to $\bar{s}$, f links from $\bar{s}$ to s, and g undirected links between s and $\bar{s}$, then

$$e + f + g = 2h$$

where h is an integer. The modified cut is then $e - f + g$, but

$$\begin{aligned} e - f + g &= (e + f + g) - 2f \\ &= 2h - 2f \\ &= 2\,(h - f) \end{aligned}$$

where h - f is an integer. So, the modified cut is also even. If it is negative, an even number of retracings will be added between s and $\bar{s}$. Therefore, it is not necessary to generate any blossom constraint since it will only require one retracing incident with a node set. The minimum modified cut will require at least two retracings if it is negative.

For an even network part b of the unicursality theorem is automatically satisfied (see section 3.2.1). The part of the algorithm investigating blossom constraints is therefore superfluous. So, the solution is achieved by repeated applications of the OKA, each of which will force an even number of retracings across some minimum modified cut if it is negative. The subroutine NONBAS must be accessed between calls to OKA to check that $\bar{\sigma} \geq 0$. We have in effect simplified the even network to a flow problem. That is also what Edmonds and Johnson (9) did and produced an algorithm which is likely to be much more efficient for this special case.

If the graph is completely directed then there is no need to grow supernodes even if there

are initial odd nodes. (See discussion in section 3.2.1 where it was shown that the evenness condition is dominated by part c of the unicursality theorem.) A modified cut will be negative for any node or set of nodes not in balance. When all nodes and node sets are balanced, then all nodes will be even. So, it suffices to balance the network and the evenness follows automatically. In terms of the algorithm the outcome is that the OKA is the only necessary column generation procedure, as was the case for an even graph.

If the network is undirected, then the balance is always satisfied. (See section 3.2.1 where it was demonstrated that the evenness condition is the only one necessary to be enforced.) So, the algorithm may omit the calls to the OKA since a column could never be generated without directed links.

The computer code checks to see if $m_1 = 1$ or $m_2 = 1$ or if the network is even. It will solve these problems as shown above. The efficiency in solving these special cases should be higher due to the simplifications that are allowed.

5.3 Aspects of Directionality

The optimal solution to the Chinese postman's problem on a mixed network must be a feasible solution to the same problem where all links are undirected. The Q blossom constraints define a convex set which must contain every optimal solution regardless of what links are directed, provided the network is connected. If all the links of the network are undirected, then part c of the unicursality theorem is automatically superfluous (see section 3.2.1), i.e., all S modified cuts are satisfied. If some of the links are made directed, then the corresponding x_{ij}'s must be non-zero. If a greater number of links are directed, then eventually there will be some modified cut which can be negative. So, with increasing m_1 on the same network there will generally be an increasing number of modified cuts which can be negative. Each one of the negative modified cuts is an additional primal constraint (hence an additional dual column), so it intersects the primal convex set defined by the blossom constraints. The algorithm will therefore require an increased effort due to the additional dual columns that are

generated.

It appears from the results that a column generated by the OKA tends to negate some explicit non-basic columns in the tableau. If intermediate output is investigated, it is quite apparent that an OKA-generated column will leave a relative large number of simplex evaluators negative. In turn, this requires an additional effort to force all explicit $z_j - c_j$'s to be non-negative. The total effort of increasing m_1 (or $\frac{m_1}{m_2}$) on execution time is therefore compounded by a greater number of OKA-generated columns each of which will cause several linear programming iterations when NONBAS is called.

CHAPTER 6

SUMMARY AND CONCLUSIONS

This study has used graph and network theory to solve a problem of routing a solid waste collection vehicle. The definition of the problem has been to find the minimum tour of a collection vehicle in a designated urban area. The premise is that the truck must collect on all the designated streets and that the collection process takes place continuously along the street rather than at discrete points. Work on related problems has preceded the present study, and the terminology, theoretical basis, and methodology from these predecessors have been used extensively. The special cases which have been solved before have been limited to street networks with either only one-way streets or only two-way streets. In most urban settings neither the two-way nor the one-way problem is a sufficient mirror of the actual traffic pattern. The present work provides a way of solving the problem on a street network with both one-way and two-way streets. The result is

an algorithm which has been translated into a computer code and which has been demonstrated to work on an actual street network.

As an exercise in network theory and operations research, the difficulty consisted of jointly modeling directed links, representing one-way street segments, and undirected links, representing two-way street segments. Separately, the two models are mathematically different so that the existing algorithms for the special cases could not be combined into a generalized algorithm. The theory for the algorithm rests on two independent theorems. One is the unicursality theorem by Ford and Fulkerson (10), the other is the matching theorem by Edmonds (5). The theorems provide a starting point for developing a linear programming formulation for the problem. However, the linear program cannot be solved by the standard simplex method due to the very large number of constraints involved, so a column generation procedure has been developed based on the dual linear program.

The column generation scheme finds non-basic dual variables that are candidates to enter the basis. In terms of the primal problem, this

procedure is equivalent to searching for violated constraints. There are three different mechanisms which may generate an entering variable. One is connected with the linear programming tableau itself and merely searches among variables that have been generated previously and have since become non-basic again. The other two are nearly independent of the linear programming tableau.

One is derived from the unicursality theorem. It operates on a network which has been derived from the original one, and applies a well-known network flow algorithm (the Out-of-Kilter Algorithm) to it. One property of the derived network is a function of the intermediate solution of the main algorithm and is provided as input from the linear programming tableau. The solution from the Out-of-Kilter Algorithm indicates whether or not a variable can be entered into the basis and how the column corresponding to that variable can be determined. This column is associated with a set of nodes which do not provide a sufficient number of exit streets and retracings to balance the number of times the truck is routed into the node set.

The last mechanism is based on the matching theorem. It monitors the original network by associating the primal linear programming variables with retracings of street segments. In so doing it is possible to determine how, if at all, any constraint of the matching theorem is violated. Such a constraint is found if there exists a node or set of nodes which is met by an odd number of street segments plus retracings. In addition it provides information about how to find the dual column corresponding to the primal constraint.

Efforts to improve the algorithm could be channeled in the direction of finding a labelling procedure. The major motivation is to develop a mechanism that is a substitute for the operation on the linear programming tableau. Other algorithms on networks and graphs have benefited from the structure of the mathematical formulation to yield a set of instructions on the nodes or links without the tableau. A labelling algorithm would likely be superior both in terms of efficiency and storage requirements.

The practical usefulness of this endeavor

can be projected along several lines. The benefit will probably not accrue from a single application of the algorithm. Urban management is rarely concerned with the design of permanent operating rules. Cities are systems which are constantly changing and evolving, and the urban manager must strive to improve the functioning of the many subsystems under these shifting conditions. The academic world can help him by developing tools as exemplified by this work. A basic justification for undertaking a study like this, is the possibility that the results of it and similar studies will be collected and made available to the urban manager. The ideal toolbox which provides ready access is the computer with remote timesharing terminals. The utility of systems analysis will not approach its potential unless its products are being used on a steady basis instead of in a single application.

A computer code to solve the Chinese postman's problem can be used to attack a variety of problems associated with solid waste collection. One case may be the temporary blocking of some streets or other changes in the traffic pattern

which affect the present routing of the collection vehicles.

A different problem is that of finding the routes for multiple trucks. This is a problem which occurs in any city of normal size. It bears a strong relationship to the problem faced in this study, but it involves another dimension--the partitioning of the city into work areas. The partitioning and the routing are interwoven and interdependent so a truly optimal algorithm would solve the two simultaneously. However, the real world of urban management has never required the universal optimal solution to any problem with which it is involved. An engineering solution is typically based on something less than perfect knowledge of the whole system, and is arrived at without considering every minute detail. In addition, simplifying assumptions normally must be made just to allow standard methods of analysis. A case in point is the multiple truck routing where the routing is done independently of the partitioning. The present algorithm lends itself ideally to assist in such a problem where it may be useful to generate a large number of possible

single truck routes in order to arrive at a desirable partitioning.

Another complex problem is one involving not only the trucks but also the depots (incinerators, landfills, transfer stations). In this case the interlocking features, which are normally considered separately, are the location of facilities, the partitioning into work areas, and the routing of the individual trucks. A problem of this type is more typical of a large city which requires more than one facility, although it only arises when new facilities are to be built. If several alternative sites are available for the new facilities, then the possible assignments of work areas to facilities may become numerous. The algorithm for the Chinese postman's problem should prove useful in generating the multitude of alternative routes based not only on present work areas but also on projected development and the attendant demand for collection service.

So, in practical terms the algorithm has its strongest promise if it is put to use as an information generator. It is hoped that it represents both a theoretical contribution to operations research and a useful aid in the hands of the urban manager.

REFERENCES

1. Bellman, Richard and K. L. Cooke, "The Konigsberg Bridges Problem Generalized," Jour. of Math. Anal. and Appl., vol. 25, pp. 1-7, 1969.

2. Berge, Claude, The Theory of Graphs, John Wiley and Sons Inc., N.Y., 1962.

3. Bodner, R. M., E. A. Cassell, and P. J. Andros, "Optimal Routing of Refuse Collection Vehicles," Jour. of the San. Eng. Div., ASCE, SA4, pp. 893-904, August 1970.

4. Edmonds, Jack, "The Chinese Postman's Problem," Bulletin of ORSA, vol. 13, Supplement 1, p. B-73, 1965.

5. Edmonds, Jack, "Maximum Matching and a Polyhedron With 0,1-Vertices," Jour. of Research, National Bureau of Standards, vol. 69B, pp. 125-130, January - June 1965.

6. Edmonds, Jack, "Paths, Trees, and Flowers," Canad. Jour. of Math., vol. 17, pp. 449-467, 1965.

7. Edmonds, Jack, "An Introduction to Matching," Mimeographed Lectures, University of Michigan Summer Eng. Conf., Ann Arbor, 1967.

8. Edmonds, Jack and Ellis L. Johnson, "Matching: A Well-Solved Class of Integer Linear Programs," in Combinatorial Structures and Their Applications, Gordon and Breach, pp. 89-92, 1970.

9. Edmonds, Jack and Ellis L. Johnson, "Matching, Euler Tours and the Chinese Postman," submitted to Mathematical Programming.

10. Ford, L. R., Jr. and D. R. Fulkerson, Flows in Networks, Princeton University Press, 1962.

11. Glover, Fred, "Finding an Optimal Edge-Covering Tour of a Connected Graph (The Chinese Postman's Problem)," ORC 67-13, Operations Research Center, University of California, Berkeley, March 1967.

12. Hadley, G., Linear Programming, Addison-Wesley Publishing Company, Inc., 1962.

13. Johnson, Ellis L., "Existence of Euler Tours in Bidirected Graphs," IBM Research, RC 2753, January 1970.

14. Kwan Mei-Ko, "Graphic Programming Using Odd or Even Points," Chinese Math., vol. 1., pp. 273-277, 1962.

15. Liebling, T. M., "Graphentheorie in Planungs- und Tourenproblemen," Lecture Notes in Operations Research and Mathematical Systems, 21, Springer-Verlag, 1970.

16. Marks, D. H. and J. C. Liebman, Discussion of "Optimal Routing of Refuse Collection Vehicles," by R. M. Bodner, E. A. Cassell and P. J. Andros, Jour. of the San. Eng. Div., ASCE, SA1, pp. 130-131, February 1971.

17. National Bureau of Standards, Unpublished Report, 1970.

18. Stricker, Robert, "Public Sector Vehicle Routing: The Chinese Postman Problem," M.S. thesis, Dept. of Electrical Eng., Mass. Inst. of Techn., Cambridge, Mass., August 1970.

19. Truitt, M. M., J. C. Liebman and C. W. Kruse, "An Investigation of Solid Waste Collection Policies," vol. 1, Dept. of Environmental Eng. Science, The Johns Hopkins University, Baltimore, Md., August 1968.

20. U.S. Department of Commerce, "1970 Census of Housing, Block Statistics, Baltimore, Md. Urbanized Area," Bureau of the Census, Washington, D.C., October 1971.